Tilt Five Unity Preview Developer Setup

Welcome to Tilt Five's developer family!

So you have an idea for an awesome Unity project with Tilt Five's hardware in mind. This guide walks you through the process of setting up your development machine in order to be a Unity developer on Tilt Five hardware.

Contents

Introduction to Tilt Five and Unity

Tilt Five utilizes a system composed of glasses, a special game board with a retroreflective surface, and a motion-tracked wand controller. The glasses project an image from each eye to the game board, which reflects back to the player's eyes to achieve a stereoscopic effect. The glasses track their position relative to the game board, as well as the position of the wand.

To use this system with a Unity project, we provide a plugin to act as a bridge between the Tilt Five hardware and Unity.

Install the Tilt Five Unity plugin in your project

Add the custom package

1. If you haven't already, now is the time to download our latest software, including the Unity plugin.
2. In Unity, create or open a project.
3. From your downloads folder, drag TiltFivePlugin-Preview.unitypackage into your Unity Project panel.
   OR
   From the Unity Assets menu, choose Import Package > Custom Package, then browse to and select the TiltFivePlugin-Preview.unitypackage file and click Open.
4. In the Import Unity Package pane, click All to select all the items. (They should be selected by default).

Creating a Tilt Five Capable Scene

Now that you have imported the plugin, it's time to display something on the glasses. There are three required components that must be present in a Tilt Five capable scene: a Game Board, a Camera to be controlled by the headset's positional data, and a Tilt Five Manager. These instructions will give you the simplest Tilt Five capable scene with the tracking origin at (0.0, 0.0, 0.0).

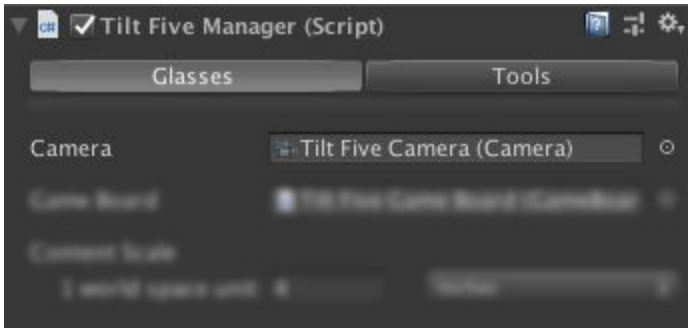
1. Create a new scene.
2. Disable or delete the Main Camera GameObject.
3. Add a cube to the scene at (0.0, 0.5, 0.0).
4. In the Project window, under Assets > Tilt Five > Prefabs, drag the Tilt Five Prototype prefab into the Hierarchy, and ensure that its Transform component has a position of (0, 0, 0) and a rotation of (0, 0, 0).
5. Press Play to run the scene in the editor.

The Tilt Five Plugin in Detail

TiltFiveManager

This is the interface to our plugin. It stores references to the current Game Board, Camera, and Wand (not in this preview), and it exposes configuration settings that manage them. Under the hood, it performs all the external library calls and setup/teardown management.

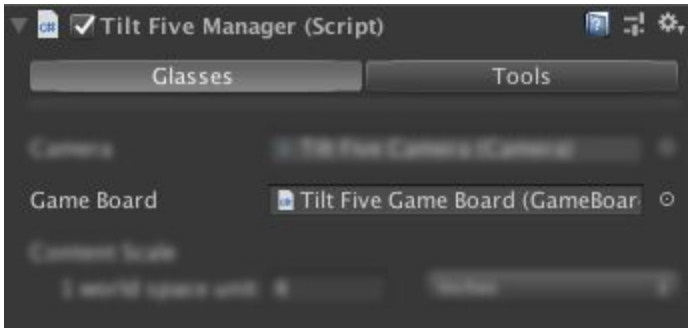
Camera



The **Camera** field points to a camera in the scene that will be controlled by the Tilt Five plugin. Since its Transform will be driven by the player's head motion, it's recommended not to directly modify the **Camera**'s Transform. Likewise, the field of view and other camera settings will be dictated by the Tilt Five plugin, so these values should also remain as they are. To overgeneralize: assign the **Camera**, and don't touch it afterwards.

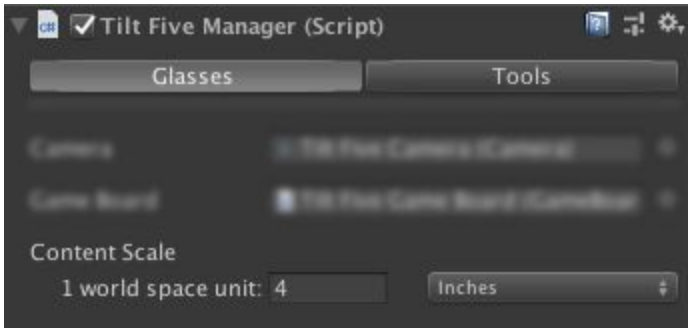
It should be noted that the **Camera** driven by the plugin will assume a default pose if the glasses are unavailable, making it impossible to move. This is also the default behavior when running the scene using the Tilt Five SDK Preview.

Game Board



The **Game Board** field points to a GameObject with the GameBoard component attached. The glasses and wand orient themselves relative to the current **Game Board**, so if you need to move the camera somewhere, it is best to do so indirectly by moving this GameObject. It is also possible to exchange one **Game Board** for another. In this way, it's possible to set up a scene with multiple GameBoard GameObjects at different positions/orientations and teleport between them.

Content Scale



The Content Scale field relates Unity's world space units to physical distances in the real world, such as meters, inches, etc. For example, if a 1x1x1 cube is placed in the scene, and the Content Scale is set to 5 centimeters, then a player wearing the glasses will see that cube as 5 centimeters wide/tall on the game board.

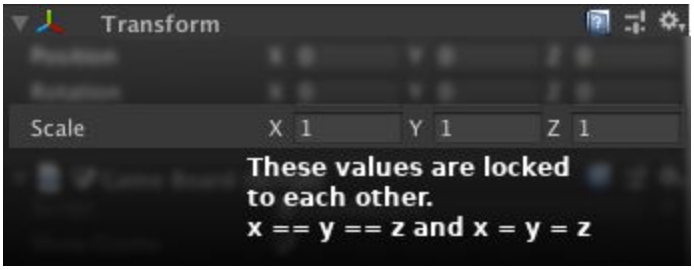
Bear in mind that Content Scale does not affect physics simulation interactions like gravity. For example, if the Unity scene being displayed is a large 1000x1000x1000 space, and the Content Scale is set to 1mm, the scene will appear to be a $1m^3$ space to the player. However, an object dropped from y1000 will take multiple seconds to fall at Unity's default gravity setting of $9.81 units/s^2$, even though it only fell 1 meter from the player's perspective. It is up to the developer to determine whether this is desirable, as this may be appropriate behavior for a large-scale simulation game, but perhaps not for a board game like chess.

It is fairly simple to increase gravity to match the player's reference frame. This can be accomplished by dividing Unity's gravity value of $9.81 units/s^2$ by the Content Scale (converted to meters), which can be accessed from the TiltFiveManager like so:

```
public class GravityCompensator : MonoBehaviour
{
  public TiltFive.TiltFiveManager tfManager;

  // It may be useful to hold onto the original gravity value.
  private Vector3 originalGravity = Physics.gravity;

  void Start()
  {
    // Convenience property for the Content Scale in Unity worldspace units.
    // (e.g. 10cm Content Scale would result in a value of 0.1)
    float gravityScalar =
tfManager.glassesSettings.physicalMetersPerWorldSpaceUnit;

    // Scaling the GameBoard will also affect gravity. To compensate:
    gravityScalar *= tiltFiveManager.glassesSettings.gameBoardScale;

    // Divide the original gravity by the Content Scale
    // (and perhaps game board scale)
    Physics.gravity = originalGravity / gravityScalar;
  }
}
```

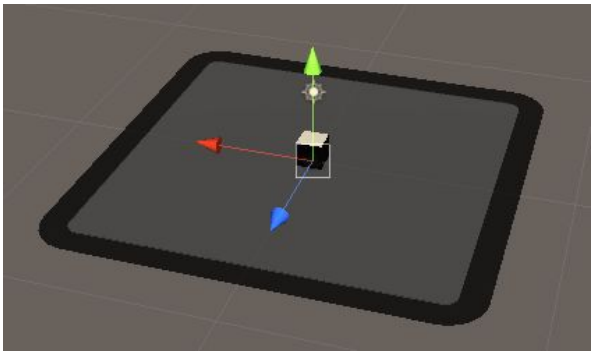
GameBoard

The **GameBoard** component uses the Transform of its GameObject to define the position, orientation, and scale of the game board. If the player's perspective is meant to move throughout a scene, this should be accomplished by moving the **GameBoard** GameObject.



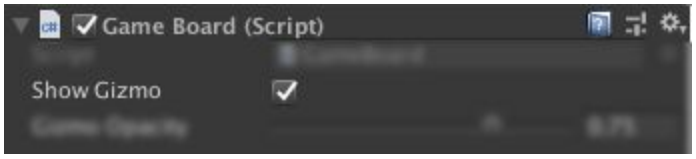
GameBoard inherits from a utility class called UniformScaleTransform, which locks the GameObject's Transform's scale axes to each other.

Note that both the TiltFiveManager's Content Scale and the **GameBoard**'s scale both affect the player's perception of scale. Content Scale is intended to be set during initialization or other transitions, whereas **GameBoard**'s scale is intended to be used for cinematic animation, such as zooms.



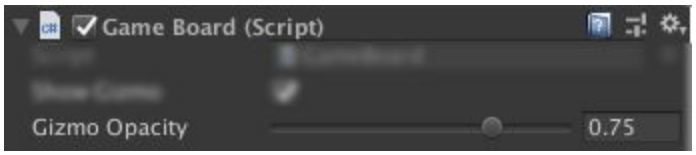
GameBoard also provides a handy gizmo to help visualize the GameBoard relative to the content in the scene.

Show Gizmo



Show Gizmo enables/disables visibility of the GameBoard gizmo. The gizmo is only visible in the Unity editor's Scene view.

Gizmo Opacity



The **Gizmo Opacity** field defines how opaque or transparent to make the gizmo.

Tilt Five Wand

This will be added in a future version of this document.